基于优先队列的时变网络最短路径算法 *

杨传印 [1,2]，黄　玮 [1,2†]，薛少聪 [1,2]，王劲松 [1,2]

(1. 天津理工大学 计算机科学与工程学院，天津 300384; 2. 天津市智能计算和软件新技术重点实验室，天津 300384)

摘　要：提出了基于优先队列的时变网络最短路径算法，能克服传统最短路径算法难以对时变网络求解最短路径的缺陷。提出的时间窗选择策略能够在算法求解过程中为节点选择合适的时间窗以降低路径长度，从而求得精确解。进一步地，算法使用了优先队列组织节点集合以提高计算效率。在随机生成的网络数据以及美国道路数据上的实验表明，基于优先队列的时变网络最短路径算法与经典方法相比，不仅能够求得精确解，运算速度也有所提高。

关键词：时变网络；优先队列；最短路径

中图分类号：TP183　　doi: 10.3969/j.issn.1001-3695.2017.12.0804

Time varying network shortest path algorithm based on priority queue

Yang Chuanyin[1,2], Huang Wei[1,2†], Xue Shaocong[1,2], Wang Jinsong[1,2]

(1. *School of Computer Science & Engineering*, *Tianjin University of Technology*, *Tianjin* 300384, *China*; 2. *Tianjin Key Laboratory of Intelligent Computing & Novel Software Technology*, *Tianjin* 300384, *China*)

Abstract: This paper presented a time-varying network shortest path algorithm based on priority queue to solve the time-varying network shortest path problem which was difficult for traditional shortest path algorithm. Proposed algorithm could address optimal solution by using proposed time-window selection strategy which could select appropriate time window for the node to reduce the path length. Also, the algorithm used the priority queue to organize node set, which could improves the computational efficiency. Experiments on randomly generated network data and united state road data show that compared with classical algorithm, the proposed algorithm is not only able to obtain global optimal solutions, but also improve the speed of the algorithm.

Key words: time varying network; priority queue; shortest path

0　引言

最短路径问题是图论和算法设计中的一个基本问题，旨在求解给定网络中给定两点之间的最短路径，在路径规划[1,2]、GPS 导航[3,4]、社交网络[5,6]等领域应用非常广泛。

在众多的最短路径算法中，Dijkstra[7]是求解单源最短路径的代表算法，其使用贪心策略能够有效求得确定网络中最短路径。但由于时间复杂度较高，Dijkstra 算法并不能在大规模网络中表现出良好的性能。为解决这一问题，学术界对其进行了大量研究。王华[8]利用邻接点算法对 Dijkstra 算法进行改进，在一定程度上降低了算法的时间复杂度和空间复杂度；马小雨等人[9]提出从减少搜索计算顶点数量入手,对经典 Dijkstra 算法进行改进，提高了算法运行的效率；王树西等人[10]针对算法无法对不连通有向图等问题进行了改进，并将其用于道路选择系统中。尽管上述改进算法具有很多优点，但其仍存在着许多约束条件，未能解决高响应、大规模环境下的网络最短路径问题。2004 年，Throup[11]使用优先队列解决确定网络下的单源最短路径问题，利用优先队列的高效性有效提高了最短路径求解的速度。

传统算法难以解决时变网络最短路径问题。时变网络与普通的静态网络不同之处在于，一点到另一点的距离随着出发时间的变化而变化。这也就使得时变网络中的最短路径问题不仅需要考虑路径中的节点顺序问题，还需要考虑节点的出发时间问题。如图 1 所示，*weight* 为走过此边所需的时间，*t* 为对应 *weight* 有效的时间区间。若求从节点 1 至节点 3 的最短路径，则有可选路径 1→2→3 和 1→3，且两条路径上分别有 4 个和 2 个时间窗。如何在算法运行过程中动态确定路径节点，并正确选择时间窗以使路径最短，是传统方法求解时变网络最短路径的原因。

时变网络比普通网络更为符合生产生活实际，在公路导航[12]、卫星网络[13]和路径规划[14]等方面都更符合实际情况。例如，在实际的道路交通环境下，一条道路在不同的时间区间下的通过时间可能是变化的，这时，使用时变网络模型来模拟道路交

收稿日期：2017-12-11；**修回日期：**2018-02-05　**基金项目：**国家自然科学基金资助项目(61673295，61301140)；天津市大学生创新创业项目(201610060063)

作者简介：杨传印（1994-），男，天津人，硕士研究生，主要研究方向为信息安全、网络通信；黄玮（1980-，男（通信作者），副教授，博士，主要研究方向为网络通信、智能计算（huangwabc@163.com)；薛少聪（1995-），女，硕士研究生，主要研究方向为网络通信；王劲松（1970-），男，研究员，博士，主要研究方向为网络安全、计算机网络.

通更加科学。学术界已对动态网络最短路径问题进行了广泛研究，并获得了一定的研究成果。戴翠琴等人[13]使用改进的 Dijkstra 算法解决卫星时变网络中的最短路径算法，邹亮等人[15]使用遗传算法解决动态网络中的最短路径问题；Sever 等人[16]使用以动态规划为基础的综合策略的方式解决变动网络中的最短路径问题。以上方法中大多需要进行多次遍历，因此存在效率低下的问题，基于启发式算法还存在求解精确度不足等问题。

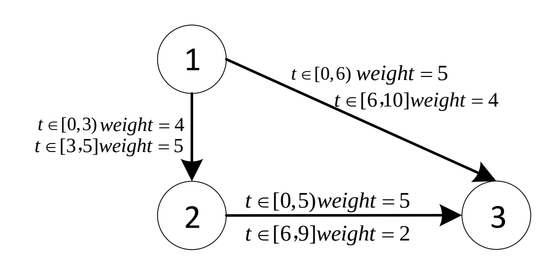


图 1　时变网络例

本文提出了一种基于优先队列的时变网络最短路径算法，算法使用时间窗选择策略，同时运用优先队列高效性的优点，能够快速求解时变网络最短路径问题。在随机时变网络数据与美国道路地图上的实验表明，提出的算法与经典算法相比，不仅能求得精确最短路径，求解速度也有所提高。

1　预备知识

优先队列[17]是一种基于堆的动态排序集合，常使用最小二叉堆为基础进行构造。利用其在数据结构上的优势，能够有效组织节点并提高计算速度。优先队列 H 支持如下操作：

$findMin(H)$：返回 H 中最小的元素。由于在优先队列中最小元素在队列首位，返回队列中第一个元素即可。此操作时间复杂度为 $o(1)$。

$insert(H,a)$:向 H 中插入元素 a。由于在插入后需要对优先队列进行维护,此操作时间复杂度为 $o(\log n)$。

$dec-key(H,a,x)$:将对象 a 的值降至 x。若 a 的值小于 x ,则返回一个错误。此操作时间复杂度为 $o(\log n)$。

$extractMin(H)$:去掉并返回 H 中最小值的元素,即队列首元素,此操作时间复杂度为 $o(\log n)$。

优先队列主要应用于基于事件驱动的模型中，一般用于对操作系统中进程的管理。操作系统将进程按优先级或执行时间进行排序后，可每次从队列首位取得需要操作的进程。得益于其较高的排序效率，在队列中某个进程的优先级发生变化时，优先队列能快速地对队列进行重新排序。

在时变网络最短路径规划问题中，由于两点之间的长度由前一个节点的出发时间决定，且一条边长的确定会影响周围多条边的长度，所以对于时变网络最短路径规划问题，需要逐个确定节点的出发时间，并在确定之后更新相邻边的长度。使用优先队列管理网络节点的方法和效果与进程管理相似：首先按节点距离路径开始节点的距离进行排序；每次取队列首位节点进行处理，即确定出发时间；在改变其他节点与开始节点的距离时，快速地重新排序。因此，利用优先队列进行时变网络最短路径规划是可行的。

2　问题定义

定义 1　时变网络（time varying network, TVN）

给定一个有向连通图 $G(V,A,W)$，其中 $V=\{v_1,v_2,...,v_n\}$ 为图中节点集合，$A=\{(v_i,v_j)\mid v_i,v_j\in V\}$ 为图 G 中边的集合，W 为一组时变函数，则称 G 为时变网络。

在时变函数 $W_{pq}(t)\in W$ 中，$W_{pq}(t)$ 为自 v_p 至 v_q 且出发时刻为 t 时，边 (v_p,v_q) 的长度。

定义 2　时间窗（time window）

在时变网络中，时间区间 $[l,u)$ 称为时间窗。其中 l 为时间窗下界，u 为时间窗上界。对于时间窗集合 $[l_1,u_1),...,[l_m,u_m)$，若 $l_1=0$ 且 $l_k<u_k=l_{k+1}$ $(k=1,...,m)$，则称此集合为时间窗集。

设边 (v_p,v_q) 上有时间窗集 tw_{pq}，tw_{pq} 包含 m 个时间窗，即 $tw_{pq}=\{tw_{pq}^i\mid i=1,...,m\}$,则在时变网络中，有

$$W_{pq}(t)=\begin{cases}\theta_1 & ,t\in tw_{pq}^1\\ ... & \\ \theta_m & ,t\in tw_{pq}^m\end{cases}\tag{1}$$

其中：θ_i 为时间 t 属于在时间窗 tw_{pq}^i 中时，边 (v_p,v_q) 的长度。

定义 3　时变网络的路径（a path of TVN）

对于有序序列 $path=<(v_1,t_1),...,(v_n,t_n)>$，$v_i$ 为序列中第 i 个节点，t_i 为 v_i 的出发时间，则称 $path$ 为时变网络的一条路径。由此也可以得出，路径 $path$ 的节点路由为节点序列 $<v_1,v_2,...,v_n>$。

时变网络最短路径求解问题即在给定的时变网络 G 中为每个节点 v 选择全局最优的出发时间 t，使自起点 S 至终点 T 的距离最小。由此，时变网络的最短路径问题可定义为

$$\begin{cases}\min & \sum_{(p,q)\in A}x_{pq}\times W_{pq}(t_i)\\ \text{subject to:} & \sum_{(1,p)\in A}x_{1q}-\sum_{(q,1)\in A}x_{q1}=1\\ & \sum_{(p,q)\in A}x_{pq}-\sum_{(q,p)\in A}x_{qp}=0\\ & \sum_{(n,q)\in A}x_{nq}-\sum_{(q,n)\in A}x_{qn}=-1\\ & x_{pq}\in(0,1),\forall(p,q)\in A\end{cases}\tag{2}$$

其中：$\sum_{(p,q)\in A}x_{pq}\times W_{pq}(t_i)$ 为最短路径长度；x_{pq} 为边 (p,q) 是否存在于网络中的标志；$W_{pq}(t_i)$ 为边 (p,q) 在 t_i 时刻出发的长度。

3　基于优先队列的时变网络最短路径算法

在时变网络中，两点之间的距离由起点的出发时间决定。因此要求解时变网络中的最短路径，应逐个确定节点的出发时间。所以，算法主要问题在于如何高效组织节点，并在对迭代过程中节点的变化及时进行维护。优先队列具有便于取出元素、排序快、维护代价低等优点。采用优先队列按与出发节点的距离的方式组织节点，便于在迭代过程中快速取得与出发节点最近的节点。在更新过程中，若某节点的情况发生变化，优先队

列可快速调整其位置，维护代价较低。因此，采用优先队列组织时变网络中的节点更为高效。

3.1　算法流程

图 2 给出了算法的流程。算法基本流程如下：首先对给定的时变网络进行处理，得到优先队列；然后进行迭代循环：为队列首节点的后置节点选择时间窗并更新其与出发节点的距离，而后移除队列首节点，直至移除了目标节点；最后从目标节点开始，沿前置节点向出发节点回溯，得到最短路径。据此，算法可分为三个部分，即优先队列初始化部分、节点更新部分及生成最短路径部分。算法的各个部分功能分述如下：

①优先队列初始化。负责对给定时变网络进行搜索以生成优先队列，并初始化节点。在节点搜索过程中，首先使用广度优先搜索算法获得有序节点序列，并为每个节点赋初值，完成优先队列及节点初始化。在广度搜索过程中，使用给定的出发节点 S 作为搜索的起点，以使 S 为队列首节点。最后对 S 进行初始化。

②节点更新。负责节点迭代更新及优先队列维护，以求出最短路径长度。每次迭代首先对队列首节点的后置节点进行时间窗选择及相关信息更新，然后将队列首节点移除并维护队列。重复此过程，直至将目标节点自优先队列移出。

③生成最短路径。在节点更新的过程中，会同时为节点记录当前最短路径中本节点的前置节点。在节点更新完成后，通过自目标节点的前置节点向出发节点递归回溯，即可得到最短路径。

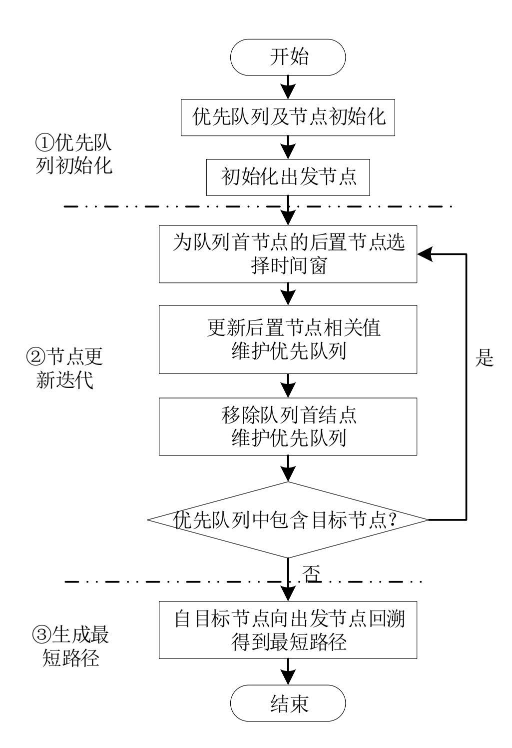


图 2　基于优先队列的时变网络最短路径算法流程

3.2　算法形式化描述

下面介绍算法的形式化描述。在开始之前，首先给出算法中的符号及符号释义，见表 1。

表 1　符号描述

符号	符号含义
G	时变网络
A	符合优先队列结构的节点集合
$BS(v)$	节点 v 的后置节点
L	已选择时间窗的节点集合
S	出发节点，即最短路径起点
T	目标节点，即最短路径终点
tw_{pq}	边 (p,q) 上的时间窗集
tw_{pq}^{i}	tw_{pq} 的第 i 个时间窗
l_{pq}^{k}	tw_{pq}^{k} 的下界
u_{ij}^{k}	tw_{pq}^{k} 的上界
θ_{pq}^{i}	选择 tw_{pq}^{i} 时边 (p,q) 的长度
F_v	v 的父节点
t_v	v 出发时间
d_v	点 v 当前最短路径长度，即与 S 的距离

基于优先队列的时变网络最短路径算法如下所示。其与流程图对应关系如下：步骤 1～3 为优先队列初始化部分；步骤 4～6 为节点更新部分；步骤 7～8 为最短路径生成部分。

算法 1　基于优先队列的时变网络最短路径算法

输入：G 、 S 、 T 。

输出：最短路径 $path$ 及最短路径长度 d_T 。

以 S 为起点，对图 G 进行广度搜索，获得包含图中全部节点的有序序列 $A=\{S,...,l_i,...,T,...,l_n\}$ ，设集合 $L=\{\varnothing\}$

对 A 中的每个节点 v 设 $d_v=\infty$ ， $F_v=\varnothing$

设 $d_{A[0]}=0$ ，即 $d_S=0$

当 $T\notin L$ 时，循环步骤 4-5：

使用**算法 2** 更新 $A[0]$ 及 $BS(A[0])$

令 $L=\{L,A[0]\}$ ，并执行 $extractMin(A)$

执行**算法 3**，得到最短路径 $path$

输出，得到最短路径 $path$ 及最短路径长度 d_T

算法 2 为节点更新算法。此子算法首先使用时间窗选择策略为节点选择时间窗，而后更新节点 q 的当前最短路径 d_q ，最后对优先队列进行维护。在算法输入部分中， $A[0]$ 为优先队列首节点， $BS(A[0])$ 为时变网络中 $A[0]$ 的后置节点。

由第 2 章定义 2 及 3 可知，时间窗集合是由一系列非负且不重叠的时间区间组成的，每个时间区间都有其对应的边长。因此在为节点选择时间窗时，首先根据前置节点的出发时间排除无效的时间窗，然后计算每个时间窗在其最早时间出发的对应的路径长度，最后在路径长度最小的时间窗中取其出发时间。

根据分析，可得出时间窗选择策略：

为 tw_{pq} 中的每个时间窗计算对应长度 $d_q^{'}$ ：若 $d_p>u_{pq}^{i}$ ，则视为此时间窗过期，令 $d_q^{'}=\infty$ ；若 $d_p\in tw_{pq}^{i}$ ，即 d_p 在时间窗中，则立即出发，令 $d_q^{'}=d_p+\theta_{pq}^{i}$ ；若 $d_p<l_{pq}^{i}$ d_p ，即 d_p 早于此时间窗，则等待 $l_{pq}^{i}-d_p$ 个单位时间，令 $d_q^{'}=l_{pq}^{i}+\theta_{pq}^{i}$ 。

为使得节点 q 能取得最短长度，若 $d_q^{'} < d_q$，则使 $d_q = d_q^{'}$，并记录当前最短路径的前置节点 $F_q = p$。

算法 2　节点更新算法

输入：$A[0]$、$BS(A[0])$、tw_{pq}、d_p、d_q。

输出：更新后的 d_q 及更新后的优先队列 A。

令 $p = A[0]$，$q \in BS(A[0])$ 对于 tw_{pq} 中的每个时间窗 tw_{pq}^i：

若 $d_p < l_{pq}^i$　，则设 $d_q^{'} = l_{pq}^i + \theta_{pq}^i$，$t_q^{'} = l_{pq}^i$

若 $l_{pq}^i \le d_p < u_{pq}^i$　，则设 $d_q^{'} = d_p + \theta_{pq}^i$　$t_q^{'} = d_p$

若 $d_p > u_{pq}^i$　，则设 $d_q^{'} = \infty$

若 $d_q > d_q^{'}$　，则设 $d_q = d_q^{'}$，$F_q = p$，$t_q = t_q^{'}$

对 A 执行 $dec - key(A, q, d_q)$

算法结束

当节点迭代完成后，集合 L 中除 S 以外的所有节点的 F_v 都已被赋值。可使用算法 3 从 T 开始向前根据 F_v 向前迭代回溯至 S 得到所求最短路径。

算法 3　路径回溯算法

输入：T、F_v、S。

输出：最短路径 $path$。

设 $q = T$，并有有序序列 $path = <q>$

当 $S \notin path$ 时，循环步骤 3-4

$q = F_q$

$path = <q, path>$

算法结束，得到最短路径路由 $path$

3.3　算法实例

为了进一步说明算法的运行过程，下面使用图 3 中的时变网络进行算例演示。图 3 中共有 5 个节点，出发节点 S 和目标节点 T 已经标出。

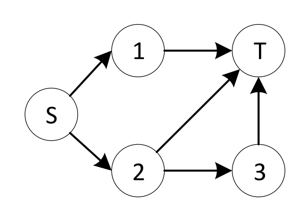


图 3　算例网络结构

时变网络的详细信息见表 2。网络中的每条边都有 1～2 个时间窗。其中时间窗为 $[0,\infty)$ 的表示不论何时出发，边的长度都为对应的 θ 值。

图 4 将算法运行过程以优先队列结构进行展示，圆内标号为节点序号，括号内为节点 d_v。图 4 中①为初始化后的优先队列结构，从②开始使用算法 2 对优先队列进行更新，⑨为最短路径长度及使用算法 3 求得的最短路径。

下面对每个步骤图进行解释：

运行算法 1 的第 1～2 步。使用广度搜索对图 G 进行搜索，生成优先队列 A，并对 A 中的节点进行初始化得到优先队列，最后初始化 S 点。

表 2　时变网络信息

节点	后置边	tw_{pq}	θ
S	（S,1）	$[0,1)$	1
		$[1,\infty)$	3
	（S,2）	$[0,2)$	1
1	（1,T）	$[0,\infty)$	5
2	（2,3）	$[0,\infty)$	3
	（2,T）	$[0,2)$	4
		$[2,\infty)$	1
3	（3,T）	$[0,\infty)$	2

对 $A[0]$ 即 S 点的邻居节点 1、2 使用算法 2 更新节点信息，其中时间窗 tw_{S1} 及 tw_{S2} 都选择了第 1 个时间窗，θ 均为 1。

将队列首元素 S 并入 L 中，并从 A 中删除，维护优先队列。此时队列首节点为节点 1。

运行算法 2 更新节点 1 的邻居节点 T 并维护优先队列。更新后 $d_T = 6$，由于 $d_T < d_3$，T 节点在队列中的位置高于节点 3。

将队列首元素节点 1 并入 L 中，从 A 中删除节点 1 并维护优先队列。此时队列首节点为节点 2。

运行算法 2 更新节点 2 的邻居节点 3、T，更新后 T 选择了 $tw_{2T}^2 = [2,\infty)$，$d_T = 3$，$d_3 = 4$。

将队列首元素节点 2 并入 L 中，并从 A 中删除。此时 T 为队列首节点。

由于 T 没有后置节点，运行算法 2 后无值更新。将队列首元素 T 并入 L 中，并从优先队列中删除。此时节点循环更新结束，得到最短路径长度 $d_T = 3$。

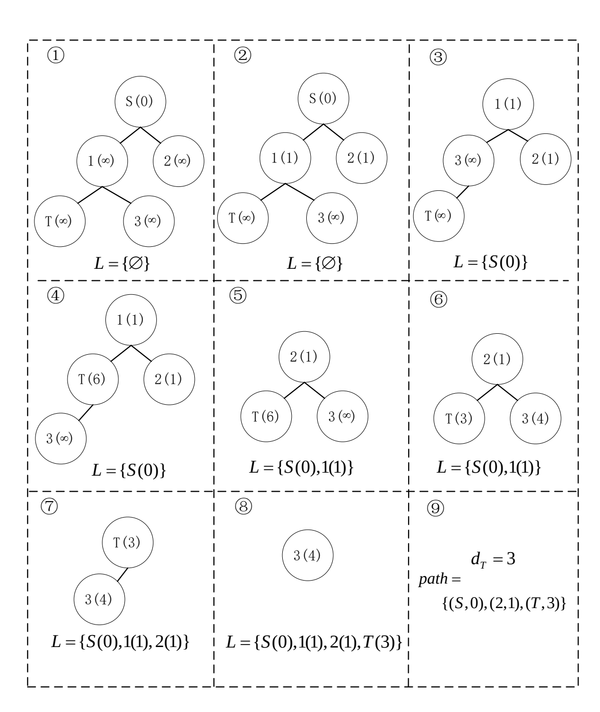


图 4　算例计算过程

使用算法 3 完成最短路径回溯，得到最短路径 $path$，算法结束。

表 3 给出了算法运行过程中每个节点各项值的变化。关注目标节点 T 在图 4 的④和⑥的两次数据变化，可以发现时间窗选择策略在寻找最优解中发挥的作用：

在④中，节点 1 为队列首元素，$d_1=1$。边 $(1,T)$ 上有唯一时间窗 $tw_{1T}^1=[0,\infty)$ $\theta_{1T}^1=5$，因此 $d_T=d_1+\theta_{1T}^1=6$。

在⑥中，节点 2 为队列首元素，$d_2=1$。边 $(2,T)$ 上有 2 个时间窗 $tw_{2T}^1=[0,2)$ $\theta_{2T}^1=4$ 和 $tw_{2T}^2=[2,\infty)$ $\theta_{2T}^2=1$，对两个时间窗分析如下：

若选择 tw_{2T}^1，即在 1 时刻立即出发，$d'_T=d_2+\theta_{2T}^1=1+4=5$；

若选择 tw_{2T}^2，即在 2 时刻出发，$d'_T=l_{2T}^2+\theta_{2T}^2=2+1=3$。

所以，选择 tw_{2T}^2，即在节点 2 处等待 1 个单位时间，并在时刻 2 出发，可以使得 d_T 达到最小值 3。

表 3　节点状态

图号	节点	d_v	F_v	tw_{pq}	所在集合
	S	0	∅	∅	A
	1	∞	∅	∅	A
①	2	∞	∅	∅	A
	3	∞	∅	∅	A
	T	∞	∅	∅	A
	S	0	∅	∅	L
	1	1	S	[0,1)	A
②-③	2	1	S	[0,2)	A
	3	∞	∅	∅	A
	T	∞	∅	∅	A
	S	0	∅	∅	L
	1	1	S	[0,1)	L
④-⑤	2	1	S	[0,2)	A
	3	∞	∅	∅	A
	T	6	1	[0,∞)	A
	S	0	∅	∅	L
	1	1	S	[0,1)	L
⑥-⑦	2	1	S	[0,2)	L
	3	4	2	[0,∞)	A
	T	3	2	[2,∞)	A
	S	0	∅	∅	L
	1	1	S	[0,1)	L
⑧	2	1	S	[0,2)	L
	3	4	2	[0,∞)	A
	T	3	2	[2,∞)	L

注：图中深色格为迭代后变化的内容。

3.4　算法分析

3.4.1 正确性分析

下面使用反证法来证明算法的正确性。由算法实例可以发现，在算法运行过程中，任意时刻时变网络中的任意一个节点一定属于集合 A 或集合 L。由此有以下定理：

定理 1　最短路径上的所有节点都属于集合 L。

证明　由图 5 所示，假设有从 S 到 T 的路径 p 和路径 p'，其中 $p'=p_1+p_2$，其中 p 中的节点全部属于集合 L，在 p' 中有至少一个节点在集合 A 中，即图中的 a 点。当算法停止时，$T\in L$ 而 $a\notin L$，说明有 $d_T\le d_a$。同时，由于图中边长皆为正值，所以有 p 的长度小于 p'。

定理 1 得证。

定理 1 从算法迭代的角度证明了最短路径中的节点均已求得当前最短路径。下面定理 2 从时间窗角度证明了时间窗选择策略能为节点选择最优时间窗以求得当前最短路径。

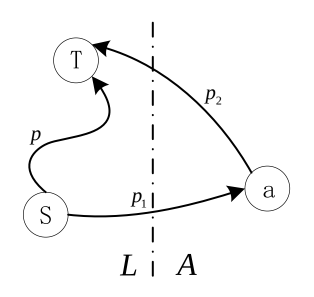


图 5　正确性说明图

定理 2　最短路径上的任意节点的出发时间所对应的当前路径长度都是最小的。

证明　设有最短路径 $path=<(v_1,t_1),...,(v_n,t_n)>$，对于任意节点 v_i，其当前路径长度 $d_i=t_i+\theta_i$。由于 θ_i 由 t_i 决定，因此 d_i 主要受 t_i 影响。下面讨论 t_i 在不同情况下对 d_i 的影响：

（1）设有 $t_i^{'}<t_i$，使得 $d_i^{'}=t_i^{'}+\theta_i^{'}$：由算法 2 的步骤 1 可知，$t_i$ 可取值可能为 d_{i-1} 或 $l_{i,i+1}$（选定时间窗下界）。若 $t_i^{'}<d_{i-1}$，根据算法 2 步骤 1.c，$d_{i+1}^{'}=\infty$；若 $t_i^{'}<l_{i,i+1}$，根据算法 2 步骤 1.d，$d_i<d_i^{'}$。因此，若 $t_i^{'}<t_i$ 则定有 $d_i<d_i^{'}$。

（2）设有 $t_i^{'}>t_i$，使得 $d_i^{'}=t_i^{'}+\theta_i^{'}$，则有以下两种情况：$t_i^{'}$ 与 t_i 在同一时间窗内，或不在同一时间窗内。若 $t_i^{'}$ 与 t_i 在同一时间窗内，则有 $d_{i+1}^{'}=t_i+\varepsilon+\theta_i$，由于 ε 为任意正数，所以 $d_i<d_i^{'}$；若 $t_i^{'}$ 与 t_i 不在同一时间窗内，根据算法 2 步骤 1.d，$d_i<d_i^{'}$。因此，若 $t_i^{'}>t_i$ 则定有 $d_i<d_i^{'}$。

定理 2 得证。

3.4.2 时间复杂度分析

定理 3　在具有 n 个节点和 m 个时间窗的时变网络中，基于优先队列的时变网络最短路径算法的时间复杂度为 $o(3n+mn+n\log(n))$。

证明　下面对三个算法进行时间复杂度分析：

算法 2：假设每个节点有 m 个时间窗，步骤 1 需要循环 m 次，步骤 2 对优先队列的维护在最差情况下需要 $o(\log(n))$ 的时间。所以算法 2 的总时间为 $m+o(\log(n))$。

算法 3：此算法是一个递归算法，用于递归求出最短路径

路由。在最坏情况下，所有节点都在最短路径上，算法 4 的时间复杂度为 $o(n)$ 。

算法 1：对节点的广度搜索及节点初始化需要遍历所有节点，即时间复杂度为 $o(n)\times 2$；初始化出发节点时间复杂度为 $o(1)$；步骤 4-6 的循环在最差情况下需要循环 n 次，所以时间复杂度为 $n\times(m+o(\log(n)))$；循环结束后的算法 4 时间复杂度为 $o(n)$ 。由此可得出算法的时间复杂度为

$$o(n)+o(n)+o(1)+n\times(m+o(\log(n)))+o(n)$$
$$=o(3n+n^2+n\log(n))\text{ 。}$$

定理 3 得证。

由定理 3 可得到如下关于时间窗数量与算法性能的推论：

推论 1 若 $m<\log(n)$，时间窗数量将不会对算法性能造成明显影响；若 $m>n$，算法的时间复杂度将显著提高。

证明 由算法的三个主要部分分别是初始化、时间窗选择、节点迭代，分别耗时 $o(2n)$ 、 $o(mn)$ 和 $o(n\log(n))$ 。

（1）若 $m<\log(n)$，则 $o(mn)<o(n\log(n))$，算法迭代的时间大于选择时间窗的时间。

（2）若 $m>\log(n)$，则 $o(mn)>o(n\log(n))$，时间窗选择的时间将超过迭代操作的时间。

（3）若 $m>n$，时间复杂度将大于 $o(3n+n^2+n\log(n))$，时间复杂度明显增长。此时，节点的时间窗复杂度会较大地影响算法的性能。

4　实验

本章将用 Dijkstra 算法、Thorup [11]的方法和基于优先队列的时变网络最短路径算法在 5 个随机生成的时变网络及美国道路网络进行实验对比。实验环境：Intel(R) Core™ i7-6700 3.4 GHz，16 GB 内存，128 GB SSD 硬盘，Windows 10 操作系统，Visual Studio 2015 编程软件，C#语言实现。

4.1　随机地图测试

本组实验数据共 5 组，节点个数分别为 1000～9000 个,使用 Randgraph（http://www.dis.uniroma1.it/challenge9/download.shtml）生成，对于地图中的每条边，参考此边的长度随机添加时间窗。表 4 概括了实验网络结构信息。

表 4　随机网络结构信息

实验数据	节点数量	边数量	时间窗数量
D_1	1000	1507	4022
D_2	3000	4479	8743
D_3	5000	7450	16871
D_4	7000	10489	22354
D_5	9000	13542	30874

每组实验都会随机选择一个出发节点和一个目标节点，进行多次重复实验。在实验过程中，基于优先队列的时变网络最短路径算法采用时间窗选择策略，其余两算法均采用选择首个可用时间窗以决定两点间的距离。

表 5　随机生成地图下的实验结果

数据编号	Dijkstra		Thorup (2004)		提出的方法	
	Error%	CT(ms)	Error%	CT(ms)	Error%	CT(ms)
D_1	10.74%	18	10.74%	7	0%	13
D_2	17.05%	136	17.05%	42	0%	46
D_3	19.88%	377	19.88%	131	0%	138
D_4	23.90%	839	23.90%	226	0%	233
D_5	23.96%	1565	23.96%	366	0%	372

表 5 为三种算法在随机网络上的实验结果，包括每种算法在不同数据集下的错误率以及运算时间（CT）。错误率为 0 表示算法达到了全局最优解。由表 5 可以看出，提出的方法由于采用了优先队列结构，在 5 组数据中求解时间增长幅度不大，符合时间复杂度预期；相比之下，Dijkstra 算法的运算时间增长迅速，在 D_5 数据集下耗时是提出算法的 4 倍；由于没有时间窗选择过程，Thourp 的运算速度较提出的方法稍快；在运算精确度方面，提出的算法采用了时间窗选择策略，可取得最优时间窗并求得最短路径，在 5 组实验中均能够能求得最优解。而 Dijkstra 和 Throup 由于直接选用首个可用时间窗，从而产生了一定且相同的误差。

4.2　美国道路地图测试

为了测试算法在真实网络中的性能，本节使用美国道路地图（http://www.dis.uniroma1.it/challenge9/download.shtml）进行测试。由于美国道路地图是静态道路网络，为了完善实验过程，实验前为网络中的部分边添加了时间窗。表 6 为实验网络结构信息。

表 6　实验用美国道路网络结构信息

实验数据	节点数量	边数量	时间窗数量
New York- D_Y	264,324	733,846	1,731,876
Colorado- D_C	435,666	1,057,066	2,621,523
Florida- D_F	1,070,376	2,712,798	6,917,634

与随机网络数据的实验方法相同，表 7 中列出了在美国道路地图中的实验结果。在大规模网络中，提出的算法依然能够求得全局最优解，而其他算法由于难以选取最优时间窗，出现了 10%～20%的计算误差。计算时间上，在美国道路地图中，Dijkstra 算法的计算时间上升最快，26 万节点的平均计算时间为 9 min，100 万节点的计算时间已经达到了约 1 h。而提出的方法在 26 万节点中的计算时间在 3 s 以下，100 万节点的平均计算时间约为 9 s，体现出优先队列结构在运算速度上的优势；时间窗选择会消耗一部分运算时间，在 D_Y 组中，提出的算法用 Thorup[11]10%的时间换取了 11.37%的准确度，在 D_F 组中用 Thorup[11]3%的额外运算时间换取了 20%的准确度。

5　结束语

本文提出了一种基于优先队列的时变网络算法用于求解时

变网络中的最短路径问题。提出的算法采用时间窗选择策略以克服传统算法难以对时变网络求最短路径精确解的问题。同时，算法采用优先队列组织网络节点，提高了算法求解速度。在随机实验数据及真实网络数据的实验表明，基于优先队列的最短路径算法能够快速求得时变网络最短路径的全局最优解。提出的算法在智能交通、路径规划及无人车等领域有更加广阔的应用前景。

表 7　美国道路地图下的实验结果

数据编号	Dijkstra		Thorup (2004)		提出的方法	
	Error%	CT(s)	Error%	CT(s)	Error%	CT(s)
D_Y	11.37%	475.7	11.37%	2.61	0%	2.95
D_C	13.48%	615.5	13.48%	3.183	0%	3.446
D_F	20.01%	3578	20.01%	8.943	0%	9.223

参考文献：

[1] 王志中. 基于改进蚁群算法的移动机器人路径规划研究 [J]. 机械设计与制造, 2018 (1): 242.

[2] 段宗涛, Wang Weixing, 康军, 等. 面向城市交通网络的 K 最短路径集合算法 [J]. 交通运输系统工程与信息, 2014, 14 (3): 194.

[3] 董俊, 黄传河. 改进 Dijkstra 算法在 GIS 导航应用中最短路径搜索研究 [J]. 计算机科学, 2012, 39 (10): 245

[4] 韩非子. 基于 GIS 的城市消防导航信息系统研究与应用 [J]. 通讯世界, 2017 (12): 253.

[5] 付东炜. 一种基于社交网络的社区关键节点的最短路径算法 [J]. 科技资讯, 2017, 15 (10): 223.

[6] Rezvanian A, Meybodi M R. Sampling social networks using shortest paths [J]. Physica A Statistical Mechanics & Its Applications, 2015, 424: 254.

[7] Dijkstra E W. A note on two problems in connexion with graphs [J]. Numerische mathematik, 1959, 1 (1): 269.

[8] 王华. 基于邻接点算法的 Dijkstra 优化研究 [J]. 计算机与数字工程, 2013, 41 (4): 518.

[9] 马小雨, 余建国. Dijkstra 算法优化及其在 GIS 中的应用 [J]. 计算机光盘软件与应用, 2014, 17 (11): 58.

[10] 王树西, 吴政学. 改进的 Dijkstra 最短路径算法及其应用研究 [J]. 计算机科学, 2012, 39 (5): 223.

[11] Thorup M. Integer priority queues with decrease key in constant time and the single source shortest paths problem [J]. Journal of Computer & System Sciences, 2004, 69 (3): 330.

[12] 唐金环, 戢守峰, 沈贵财. 时变网络下考虑碳排放的车辆路径优化 [J]. 系统工程, 2015, 33 (9): 37.

[13] 戴翠琴, 李剑, 唐煌. 卫星时变网络中基于连接计划的最短路径优化算法 [J]. 重庆邮电大学学报: 自然科学版, 2017, 29 (01): 29.

[14] 胡腾波, 叶建栲. GIS 时变权值网络最短路径算法研究 [J]. 计算机与现代化, 2008 (11): 22.

[15] 邹亮, 徐建闽. 基于遗传算法的动态网络中最短路径问题算法 [J]. 计算机应用, 2005 (4): 742.

[16] Sever D, Dellaert N, Van Woensel T, *et al.* Dynamic shortest path problems: Hybrid routing policies considering network disruptions [J]. Computers & Operations Research, 2013, 40 (12): 2852.

[17] Thomas H. Cormen, introduction to algorithms. [M]. 3rd ed. Massachusetts: The MIT Press, 2009: 183.